View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Elsevier - Publisher Connector

JOURNAL OF COMPLEXITY, **13,** 42–49 (1997)
ARTICLE NO. CM970438

Rectangular Matrix Multiplication Revisited

Don Coppersmith

IBM Research, T. J. Watson Research Center, Yorktown Heights, New York 10598

Received May 28, 1996

DEDICATED TO SHMUEL WINOGRAD ON HIS 60TH BIRTHDAY

We give a constant $\alpha > 0.294$ and, for any $\varepsilon > 0$, an algorithm for multiplying an $N \times N$ matrix by an $N \times N^\alpha$ matrix with complexity $O(N^{2+\varepsilon})$. ©1997 Academic Press

1. INTRODUCTION

In 1969 Strassen [6] showed how to multiply a pair of 2×2 matrices with a bilinear algorithm using seven multiplications instead of eight, and from that developed an algorithm for multiplying a pair of square $N \times N$ matrices using $O(N^\omega)$ operations where $\omega = \log 7/\log 2 \approx 2.807$.

Since then many improvements have been made for the case of square matrices, the latest being [3], which reduces the estimate of ω to 2.376. See Pan's book [4] for some of the history of this field.

The present author [2] investigated the complexity of multiplying rectangular matrices and showed that, for a constant $\alpha > 0.172$ and for any $\varepsilon > 0$, there is an algorithm for multiplying a square $N \times N$ matrix by a rectangular $N \times N^\alpha$ matrix, with complexity $O(N^{2+\varepsilon})$. Since the square matrix has N^2 elements, the complexity has a lower bound of N^2, so that this result is extremely close to its lower bound.

In the present note we combine ideas from this earlier work on rectangular matrices [2] with ideas from the latest work on square matrices [3] to improve the constant α in this result. Namely, we show that, for a constant $\alpha > 0.294$ and for any $\varepsilon > 0$, there is an algorithm for multiplying a square $N \times N$ matrix by a rectangular $N \times N^\alpha$ matrix, with complexity $O(N^{2+\varepsilon})$.

0885-064X/97 $25.00
Copyright ©1997 by Academic Press
All rights of reproduction in any form reserved.

2. NOTATION

The problem of multiplying an $m \times n$ matrix by an $n \times p$ matrix to produce an $m \times p$ matrix will be denoted by $\langle m, n, p\rangle$. Indices i, j, k have ranges of size m, n, p, respectively.

The notation $L \to \langle m, n, p\rangle$ indicates the existence of a trilinear algorithm requiring L essential multiplications to compute the indicated matrix product. If the algorithm is an "approximate algorithm" [1] we write $L \xrightarrow{\lambda} \langle m, n, p\rangle$. If k disjoint matrix products are computed (sharing no variables), we write $L \to k\langle m, n, p\rangle$.

3. BASIC ALGORITHM

Much of the theory necessary for understanding this result is developed in the papers [2, 3] and in Pan's book [4]. We begin with a basic algorithm from [3, Equation (10)]. For a given value of the integer q (=6 or 7) we will call this construction D_q.

$$\begin{aligned}
&\sum_{i=1}^{q} \lambda^{-2}(x_0^{[0]} + \lambda x_i^{[1]})(y_0^{[0]} + \lambda y_i^{[1]})(z_0^{[0]} + \lambda z_i^{[1]}) \\
&\quad - \lambda^{-3}\left(x_0^{[0]} + \lambda^2 \sum_{i=1}^{q} x_i^{[1]}\right)\left(y_0^{[0]} + \lambda^2 \sum_{i=1}^{q} y_i^{[1]}\right)\left(z_0^{[0]} + \lambda^2 \sum_{i=1}^{q} z_i^{[1]}\right) \\
&\quad + [\lambda^{-3} - q\lambda^{-2}](x_0^{[0]} + \lambda^3 x_{q+1}^{[2]})(y_0^{[0]} + \lambda^3 y_{q+1}^{[2]})(z_0^{[0]} + \lambda^3 z_{q+1}^{[2]}) \\
&= \sum_{i=1}^{q} (x_0^{[0]} y_i^{[1]} z_i^{[1]} + x_i^{[1]} y_0^{[0]} z_i^{[1]} + x_i^{[1]} y_i^{[1]} z_0^{[0]}) \\
&\quad + x_0^{[0]} y_0^{[0]} z_{q+1}^{[2]} + x_0^{[0]} y_{q+1}^{[2]} z_0^{[0]} + x_{q+1}^{[2]} y_0^{[0]} z_0^{[0]} + O(\lambda).
\end{aligned}$$

This is an "approximate algorithm" in the sense of [1], with λ playing the role of an infinitesimal; calculations will be done in the ring of formal polynomials in $\{\lambda, \lambda^{-1}\}$, introducing a multiplicative factor of $\log^2(N)$ to the complexity, which can be hidden in the N^{ε} error term.

This trilinear algorithm uses q + 2 multiplications (the left-hand side) to compute six different matrix products (the right-hand side). The number of x-variables is also q + 2. This agreement is necessary to achieve the near equality between the number of operations and the number of x-variables in the larger algorithm.

The six matrix products share variables in an incompatible manner; for example, the second and third products both involve $x_i^{[1]}$, the second as a column vector, the third as a row vector. One of the main contributions of [3], inspired in turn by [7], is to deal with this incompatibility.

Each of the three sets of variables (x, y, and z) is considered to be broken into three "blocks": $x^{[0]} = \{x_0^{[0]}\}$, $x^{[1]} = \{x_1^{[1]}, x_2^{[1]}, \ldots, x_q^{[1]}\}$, and $x^{[2]} = \{x_{q+1}^{[2]}\}$, and similarly with y and z. These block numbers are assigned in such a way that variables contained in blocks $x^{[i]}$, $y^{[j]}$, and $z^{[k]}$ appear together in a matrix product if and only if $i + j + k = 2$.

4. ITERATION

Following [3], we are going to take a large tensor power of this construction, in order to deal with the incompatible repetitions of blocks of variables. We will use Salem and Spencer's [5] construction of a large set of integers free of three-term arithmetic progressions to eliminate certain blocks of variables, and to insure that each remaining block is contained in only one matrix product. We also borrow from [2] the idea of selecting multinomial coefficients to maximize the total number of x-variables in the resulting product, so that the total number of operations is not much more than the number of x-variables. This is made possible by the fact that in the original construction D_q the number of multiplications is the same as the number of x-variables, namely q + 2.

Let K be a large even integer. Set $L = [0.6425K]$, $\beta = 0.010156$, $B = [\beta K]$, $\gamma = 0.01452$, and $C = [\gamma L]$, where $[*]$ denotes the greatest integer function.

Take the tensor product of $9K$ copies of construction D_7 and $8L$ copies of construction D_6.

The total number of multiplications involved is $9^{9K}8^{8L}$.

Each variable has $9K$ + $8L$ indices: $9K$ on the left, from the q = 7 part, and $8L$ on the right, from the q = 6 part.

We will set many blocks of variables to 0. Of the x-variables, we retain only those whose left-hand $9K$ indices contain exactly K from block 0, $7K$ from block 1, and K from block 2, and whose right-hand $8L$ indices contain exactly L from block 0, $6L$ from block 1, and L from block 2. The number of blocks of x-variables retained is then

$$A = \binom{9K}{K,\,7K,\,K}\binom{8L}{L,\,6L,\,L} = \left[\left(\frac{9^9}{7^7}\right)^K \left(\frac{8^8}{6^6}\right)^L\right]^{1-o(1)},$$

where the notation $\binom{a}{b,c,d}$ indicates a multinomial coefficient. Each block of x-variables contains $7^{7K}6^{6L}$ variables, so that the total number of x-variables is ap-

proximately the same as the number of multiplications, that is, $(9^{9K}8^{8L})^{1-o(1)}$. (The closeness of the approximation is indicated by the exponent $1 - o(1)$.)

Of the y-variables and the z-variables, we retain those whose left-hand indices contain $9K/2 + B$ from block 0, $9K/2 - 2B$ from block 1, and B from block 2, and whose right-hand indices contain $4L + C$ from block 0, $4L - 2C$ from block 1, and C from block 2. The number of blocks of y-variables is then

$$\begin{aligned}\binom{9K}{\frac{9K}{2}+B,\frac{9K}{2}-2B,B}\binom{8L}{4L+C,4L-2C,C}&=\Bigg[9^{9K}\left(\frac{9}{2}+\beta\right)^{-(9/2+\beta)K}\left(\frac{9}{2}-2\beta\right)^{-(9/2-2\beta)K}\beta^{-\beta K}\\&\quad\times 8^{8L}(4+\gamma)^{-(4+\gamma)L}(4-2\gamma)^{-(4-2\gamma)L}\gamma^{-\gamma L}\Bigg]^{1-o(1)}.\end{aligned}$$

We have chosen β, γ, L, and K to make the number of y-blocks approximately equal to the number of x-blocks. The number of z-blocks is the same.

For each matrix product that remains, the left-hand part will consist of a tensor product of some permutation of the following: K copies of $x_8^{[2]}y_0^{[0]}z_0^{[0]}$ (a matrix product of size $\langle 1, 1, 1\rangle$), $7K/2$ copies of $\sum x_i^{[1]}y_i^{[1]}z_0^{[0]}$ ($\langle 1, 7, 1\rangle$), $7K/2$ copies of $\sum x_i^{[1]}y_0^{[0]}z_i^{[1]}$ ($\langle 7, 1, 1\rangle$), B copies of $x_0^{[0]}y_8^{[2]}z_0^{[0]}$ ($\langle 1, 1, 1\rangle$), $K - 2B$ copies of $\sum x_0^{[0]}y_i^{[1]}z_i^{[1]}$ ($\langle 1, 1, 7\rangle$), and B copies of $x_0^{[0]}y_0^{[0]}z_8^{[2]}$ ($\langle 1, 1, 1\rangle$). The number of copies of $x_8^{[2]}y_0^{[0]}z_0^{[0]}$ is determined by the number of x-indices in block 2, namely K, and similarly by permuting indices; then the number of copies $\sum x_0^{[0]}y_i^{[1]}z_i^{[1]}$ is determined by the number of x-indices in block 0 not used for $x_0^{[0]}y_0^{[0]}z_8^{[2]}$ or $x_0^{[0]}y_8^{[2]}z_0^{[0]}$. In the same manner we see that the right-hand part will consist of a tensor product of: L copies of $x_7^{[2]}y_0^{[0]}z_0^{[0]}$ ($\langle 1, 1, 1\rangle$), $3L$ copies of $\sum x_i^{[1]}y_i^{[1]}z_0^{[0]}$ ($\langle 1, 6, 1\rangle$), $3L$ copies of $\sum x_i^{[1]}y_0^{[0]}z_i^{[1]}$ ($\langle 6, 1, 1\rangle$), C copies of $x_0^{[0]}y_7^{[2]}z_0^{[0]}$ ($\langle 1, 1, 1\rangle$), $L - 2C$ copies of $\sum x_0^{[0]}y_i^{[1]}z_i^{[1]}$ ($\langle 1, 1, 6\rangle$), and C copies of $x_0^{[0]}y_0^{[0]}z_7^{[2]}$ ($\langle 1, 1, 1\rangle$). So the product will have size $\langle Q, Q, R\rangle$, where

$$\begin{aligned}Q&=7^{7K/2}6^{3L}\\R&=7^{K-2B}6^{L-2C}=Q^{\alpha}\\\alpha&=0.29462\ldots.\end{aligned}$$

The number of y-variables will be substantially smaller than the number of x-variables, even though the number of blocks is the same. This is because of the way that we selected the indices of those variables which we retain; they were selected to maximize the total number of x-variables retained, but not y-variables.

We say that a given three blocks of variables, an x-block, a y-block, and a z-block, are "compatible" if they appear together in one of the matrix products. This condition is equivalent to the condition that the block indices add up to 2 in each of the $9K + 8L$ positions.

Among the surviving blocks (those not set to 0), the number of pairs of y-blocks and z-blocks compatible with a given x block is given by the product

$$M' = \binom{K}{K}\binom{7K}{\frac{7K}{2},\frac{7K}{2}}\binom{K}{B,K-2B,B}\binom{L}{L}\binom{6L}{3L,3L}\binom{L}{C,L-2C,C}.$$

5. ARITHMETIC PROGRESSIONS

Following [3], we define a modulus $M = 2M' + 1$. The Salem–Spencer Theorem [5] shows how to construct a subset $P \subseteq \mathbf{Z}/M$ with $|P| = M^{1-o(1)}$, free of three-term arithmetic progressions: if $a, b, c \in P$ and $a + c = 2b \pmod M$ then $a = b = c$. We will use this subset to selectively set more blocks of variables to 0, thereby eliminating duplicate (incompatible) uses of the surviving blocks of variables.

We randomly select weights $w_i \in \mathbf{Z}/M$, $1 \le i \le 9K + 8L$. An x-block X with block indices $a_i \in \{0, 1, 2\}$, $1 \le i \le 9K + 8L$, is assigned a weight $a(X) = \sum_i a_i w_i$. Similarly, a z-block Z with block indices c_i is assigned a weight $c(Z) = \sum_i c_i w_i$. A y-block is treated a little differently: a y-block Y with block indices b_i is assigned a weight $b(Y) = \frac{1}{2}\sum_i (2 - b_i) w_i$. Because M is odd, multiplication by $\frac{1}{2}$ is well defined.

A compatible triple of blocks (X, Y, Z) satisfies $a_i + b_i + c_i = 2$, $1 \le i \le 9K + 8L$. This translates into the condition $a(X) + c(Z) = 2b(Y)$.

We retain those blocks X with $a(X) \in P$, and similarly require $b(Y) \in P$ or $c(Z) \in P$ for retention of blocks Y or Z. Blocks not retained are set to 0.

Now the only matrix products remaining are those for which the block indices satisfy $a(X) + c(Z) = 2b(Y)$, while $a(X), b(Y), c(Z) \in P$. By the Salem–Spencer property, this implies $a(X) = b(Y) = c(Z)$. For a given x-block, among the M' y-blocks compatible with it (and the z-blocks, uniquely determined by the x- and y-blocks), the expected number of blocks that will have the correct value $b(Y) = a(X)$ is $M'/M \approx \frac{1}{2}$. Because the numbers of x-blocks, y-blocks, and z-blocks are roughly equal, we can make a similar statement after permuting the variables.

Some x-blocks will be associated with two or more surviving y-blocks. When this happens we set to zero all but one of these y-blocks. Do a similar procedure whenever a y-block has two or more x-blocks associated with it, or similarly with a z-block.

We began with A blocks X_s. The Salem–Spencer construction reduced this number to $A \times M^{-o(1)} = A^{1-o(1)}$. This number is further reduced, because of two or more surviving y-blocks associated with a given x-block, or because an x-block has no y-blocks associated with it, but when we are finished the number A' of surviving triples of blocks (X_s, Y_s, Z_s) still satisfies $A' = A^{1-o(1)}$. Each triple is compatible, and thus represents a matrix product that is computed by our algorithm. Any other surviving blocks (X_s, Y_t, Z_u) of variables are not compatible unless $s = t = u$. So the several matrix products are "disjoint": they share no variables.

Each surviving triple represents a matrix product of size $\langle Q, Q, R\rangle$, where $Q = 7^{7K/2}6^{3L}$ denotes the number of indices shared by x and z variables (the i-indices), which is equal to the number of indices shared by x and y variables (the j-indices), and $R = 7^{K-2B}6^{L-2C}$ denotes the number of indices shared by y and z variables (the k-indices).

So we have given an approximate algorithm for:

$$9^{9K}8^{8L} \xrightarrow{\lambda} A'\langle Q, Q, R\rangle.$$

The right-hand side is the disjoint sum of A' different matrix products. The left-hand side is

$$9^{9K}8^{8L} = A^{1+o(1)}Q^2 < A'Q^{2+\varepsilon/2},$$

the latter estimate holding when K is sufficiently large. Thus

$$A'Q^{2+\varepsilon/2} \xrightarrow{\lambda} A'\langle Q, Q, R\rangle.$$

Now we iterate this construction S times for some sufficiently large integer S:

$$\begin{aligned} A'Q^{S(2+\varepsilon/2)} = A'Q^{2+\varepsilon/2}Q^{(S-1)(2+\varepsilon/2)} &\xrightarrow{\lambda} A'Q^{(S-1)(2+\varepsilon/2)}\langle Q, Q, R\rangle \\ &\xrightarrow{\lambda} A'Q^{(S-2)(2+\varepsilon/2)}\langle Q^2, Q^2, R^2\rangle \\ &\xrightarrow{\lambda} \cdots \\ &\xrightarrow{\lambda} A'\langle Q^S, Q^S, R^S\rangle \\ &\to \langle Q^S, Q^S, R^S\rangle. \end{aligned}$$

For S sufficiently large we will have $A' < Q^{\varepsilon S/4}$, implying

$$Q^{S(2+3\varepsilon/4)} \xrightarrow{\lambda} \langle Q^S, Q^S, R^S \rangle.$$

Recall that the λ indicates an "approximate algorithm." This is taken care of by letting each multiplication on the left represent a multiplication of polynomials in $\{\lambda, \lambda^{-1}\}$, whose cost is at worst the square of the degree of the polynomials. This degree in turn is logarithmic in Q^S, and so can be absorbed into a $Q^{\varepsilon S/8}$ factor, yielding

$$Q^{S(2+7\varepsilon/8)} \to \langle Q^S, Q^S, R^S \rangle.$$

Thus, setting $N_0 = Q^S$ we get

$$N_0^{2+7\varepsilon/8} \to \langle N_0, N_0, N_0^{\alpha} \rangle,$$

where

$$\alpha = \frac{\log R^S}{\log Q^S} = \frac{\log R}{\log Q} = 0.29462\ldots.$$

This holds for one specific value of N_0. By taking tensor powers again and approximating, we can make this equation valid for all sufficiently large N, as well as incorporating all operations (not just essential multiplications) into the complexity estimate, at the cost of another $N^{\varepsilon/8}$. Finally, we have:

Theorem 1. *Let α = 0.29462…. For all $\varepsilon > 0$, for all sufficiently large N, there is an algorithm for multiplying an $N \times N$ matrix by an $N \times N^{\alpha}$ matrix at the cost of $N^{2+\varepsilon}$ arithmetic operations.*

REFERENCES

1. D. Bini, M. Capovani, G. Lotti, and F. Romani, $O(n^{2.7799})$ complexity for matrix multiplication, *Inform. Process. Lett.* **8** (1979), 98–108.
2. D. Coppersmith, Rapid multiplication of rectangular matrices, *SIAM J. Comput.* **11,** No. 3 (Aug. 1982), 467–471.

3. D. Coppersmith and S. Winograd, Matrix multiplication via arithmetic progressions, *J. Symbolic Comput.* **9** (1990), 251–280.
4. V. Ya. Pan, "How to Multiply Matrices Faster," Springer Lecture Notes in Computer Science, Vol. 179, Springer-Verlag, New York/Berlin, 1984.
5. R. Salem and D. C. Spencer, On sets of integers which contain no three terms in arithmetical progression, *Proc. Natl. Acad. Sci. USA* **28,** 561–563.
6. V. Strassen, Gaussian elimination is not optimal, *Numer. Math.* **13** (1969), 354–356.
7. V. Strassen, Relative bilinear complexity and matrix multiplication, preprint, 1986; see also, The asymptotic spectrum of tensors and the exponent of matrix multiplication, *Proc. 27th Annual IEEE Symposium on Foundations of Computer Science,* 1986, pp. 49–54.